## REMARKS

This Amendment is in response to the Final Office Action mailed October 13, 2010 and is submitted subsequent to the telephone interview of Feb. 1, 2011, for which the undersigned thanks Exr. McClellan.

During the interview, Exr. McClellan indicated that the arguments below appeared to be persuasive and that the present amendment would likely be entered, but cautioned the undersigned that a further search would likely, in his opinion, reveal additional art relevant to the claims. During a subsequent telephone call, Exr. McClellan indicated that the undersigned should address his concerns that Mockapetris' "software ring" configuration (middle paragraph of page 152 of Mockapetris) may read on the claims as now presented.

Claim 1 has been amended to take care of a typo and claim 79 has been amended as to form. Neither of these amendments is of a nature which would preclude entry of the present amendment after final.

Claims 1-5, 8-12, 63-65 and 108 were rejected as being unpatentable over Mockapetris in view of Nguyen. Claim 109 was further rejected as being unpatentable over the Mockapetris-Nguyen combination, in further view of San Andres. Reconsideration and withdrawal of these rejections are respectfully requested.

With regard to Mockapetris' statement on page 152, Column 2 that an important goal of his system is to "optimize the multicast potential of the medium without incurring excessive cost in terms of processing events in the receivers of the distribution" and the Examiner assertion that "With that goal explicit, a single acknowledgment would be sufficient to complete a transaction", the undersigned respectfully submits the following.

Mockapetris seeks to optimize the multi-casting of the messages without incurring excessive costs in terms of processing events in the receivers of the distribution, by improving the probability that the transmissions are successful and discarding duplicate **transmissions** (right-hand col., page 152):

> The first of these is optimizing the performance of packet primitives in the network interface. Our goal is to optimize the multicast potential of the medium without incurring excessive cost in terms of processing events in the receivers of the distribution. This goal is achieved through measures to improve the probability that transmissions are successful and measures to rapidly discard irrelevant or duplicate transmissions. In this regard, multicast is more sensitive to the effects of errors than one-to-one transmission because although a failure may still double the cost, the cost of multicast increases with the size of the multicast set.

Therefore, Mockapetris seeks to 1) optimize the multi-cast transmission of the messages (in a manner that does not impose excessive burdens on the receivers thereof) and 2) to do so by ("This goal is achieved through...") a) improving the probability that the transmissions are successful and b) discarding irrelevant or duplicate transmissions. These, then are the outgoing transmissions of the multicast messages.

These goals of maximizing the **sending of the messages** are discussed in the "Packet primitives strategies and costs" section of Mockapetris:

> **Packet primitives strategies and costs Several techniques for improving interface performance are already in use in various systems. Interfaces should recognize multicast addresses instead of a single broadcast address; thus hosts that are not in the multicast set won't have to expend time to discard extraneous packets. Network interfaces can minimize packet loss through full duplex operation and by buffering strategies that allow reception of back-to-back packets from the medium. A fairly simple extension to this scheme would be to reserve a buffer for each multicast connection so that multicast distributions would never be discarded due to lack of resources.**

> **In situations such as saturation, where the ACK reliability isn't a problem, but the distribution may still need retransmission, duplicate detection can be made automatic by using two multicast addresses and a variation on the alternating bit protocol[12]. In this scheme, called *parity*, the sender uses one address for the initial transmission *and all retransmissions* of a message, and then switches to the other address for the next message. The receivers change addresses whenever they successfully copy a new message. Receivers that miss a message stay with the old address and eventually receive a retransmission; receivers that copy a message are spared receiving any future retransmissions. The parity system requires restrictions on the packet lifetime and outstanding messages that are rarely a problem in a local network. The ultimate in performance is achieved by a network interface that performs duplicate detection, ACK generation, and ACK reception without host intervention. These interfaces are referred to as *filter* interfaces in further discussion. The parity scheme is a primitive example of automatic duplicate detection; various interfaces, such as the Hyperchannel[13] and others[14] incorporate automatic ACK generation, though at a low level in the protocol hierarchy. A scheme for a high-level version is described by Mockapetris[11]. These acknowledgments are transmitted immediately following the distribution they acknowledge, and hence are called *prompt* ACKs.**

These passages make no mention, teaching or suggestion of ...

> **"the outbound game payload enabling the gaming machine having sent the transaction packet to complete the game transaction and wherein the at least one gaming machine is configured such that a first arriving outbound payload received by the at least one gaming machine is effective to complete the game transaction, irrespective of when and if a second later arriving outbound payload is received by the at least one gaming machine."**

... as claimed in claim 1 and as similarly claimed in the other pending independent claims. Indeed, Mockapetris teaches for the interface to recognize multicast addresses instead of a single broadcast address and a *parity* scheme for duplicate **transmission** detection. So far, in Mockapetris, we are only talking about the initial transmission of the multicast messages, and have not yet reached the point where the ACKs are generated by the hosts (recipients of the multicast messages). Here, there is not teaching or suggestion therein for a sender to treat a first arriving outbound payload received by the at least one gaming machine as being effective to complete the game transaction, irrespective of when and if a second later arriving outbound payload is received by the at least one gaming machine, as claimed herein.

Mockapetris also wishes to optimize the acknowledgment algorithm (the generation and sending of the ACKs at the receiving end of the multi-cast transmission back to the original multicast message sender) and the associated burden that such acknowledgements pose at the recipients thereof (the original senders referred to above):

> We also want to optimize the acknowledgment algorithm. In multicast, there is more distinct acknowledgment information than data to be acknowledged; hence special acknowledgment algorithms may be justified.

Mockapetris does this through one of four multicast algorithms:

1. **Simulation algorithms;**

2. **Multiple acknowledgment algorithms;**

3. **Saturation algorithms, and**

4. **Negative acknowledgment algorithms.**

These algorithms are concerned with the **cost of sending of acknowledgments** by the hosts to the sender of the original multicast message.

> **Acknowledgment strategies and costs**
>
> The focus for acknowledgments is eliminating the cost of ACK transmissions, either by moving the cost into the network interface or by reducing the number of ACKs required. Four types of multicast algorithms are considered:

Even when Mockapetris states that "there is more distinct acknowledgement information than data to be acknowledged", he does not teach that a first arriving outbound payload received by the at least one gaming machine is treated as being effective to complete the game

transaction, irrespective of when and if a second later arriving outbound payload is received by the at least one gaming machine, as claimed herein. This is simply because such a scheme would be contrary to the purpose of multicasting messages, as defined by Mockapetris: to insure that all intended recipients of the message receive the message. In contrast, the claimed embodiments only require a single outbound payload to be returned (irrespective of how many transaction packets were sent to the central servers) to complete the game transaction. How many others are subsequently received (if any), does not affect the completion of the transaction, according to the claimed embodiments.

Contrary to Mockapetris, the claimed embodiments are, again, unconcerned with the costs of ACK transmissions by either moving the burden to a network interface of by reducing the number of ACKs required.

Indeed, in the claimed embodiments, irrespective of the number of ACKS generated by the central servers, **a first arriving outbound payload received by the at least one gaming machine is treated as being effective to complete the game transaction, irrespective of when and if a second later arriving outbound payload is received by the at least one gaming machine**, as claimed herein. Mockapetris simply does not teach or suggest that the sender of the multicast message, after having sent same, discards all but the first-to-arrive ACK received from the ACK-sending hosts.

At no point is Mockapetris believed to teach or suggest the utility or desirability of discarding ACKs, after a first in time ACK has been received by the sender of the multicast. The whole point of multicasting is the transmission of a message to predetermined multiple receivers and to ensure that the intended receivers thereof actually received the multicast messages:

In the **Simulation Algorithm** (number 1 above), each host sends an ACK back to the sender or a software ring is used, in which the last destination acknowledges receipt to the sender. To address the Examiner's concern regarding the software ring and the present claims, the undersigned submits the following.

Mockapetris states that, in the "software ring" configuration, "the source transmits the message to the first destination, which forwards the message to the next destination, etc. The last destination returns the message or an ACK to the sender. ... The advantages of this scheme are the reduction in traffic and the sender needn't maintain a list of all destinations; each member need only remember the next member".

This is believed to be incompatible with the claimed language:

> **each of the at least one gaming machine being configured to play at least one game and to carry out a game transaction for each game played and to commit each game transaction to each of the at least two central servers by sending a single transaction packet to each of the at least two central servers, each single transaction packet sent to each of the at least two central servers including an identical inbound game payload wherein each of the at least two central servers, upon receipt of the inbound game payload, are configured to return a single outbound game payload to the gaming machine having sent the transaction packet**

Indeed, in the "software ring", the sender does not carry out any step of "sending a single transaction packet to each of the at least two central servers". Mockapetris itself supports this interpretation, as the reference explicitly states that "the source transmits the message to the first destination ... each member need only remember the next member". Moreover, the claim states:

> **...each single transaction packet sent to each of the at least two central servers including an identical inbound game payload...**

In a ring configuration, there is no each single transaction packet sent to each of the at least two servers including an identical inbound game payload. The claim recites at least two

central servers and recites that each single transaction packet sent to each of the at least two central server includes an identical game payload. This language is substantively and grammatically inconsistent with a single game payload being forwarded to successive destinations. In contrast, in Mockapetris' "software ring", there are no identical inbound payloads, there is only one payload, that gets forwarded along the ring, to each successive destination.

Continuing through Claim 1, the claim then recites:

> **...wherein each of the at least two central servers, upon receipt of the inbound game payload, are configured to return a single outbound game payload to the gaming machine having sent the transaction packet...**

This too is inconsistent with Mockapetris' "software ring" implementation, in that the destinations, upon receipt of the message, do not (except for the last one in the ring), return a single outbound game payload to the gaming machine having sent the transaction packet. What does happen, in Mockapetris, is that each destination forwards the message to the next member of the ring, and not to the gaming machine having sent the transaction packet. That the last one in the chain does so is not fatal to this argument, as the claim requires that each of the at least two servers, upon receipt of the inbound payload, are configured to return a single outbound game payload to the gaming machine having sent the transaction packet (as claimed), and not to the "next member" of the "software ring", as taught by Mockapetris.

Lastly, claim 1 recites

> **...wherein the at least one gaming machine is configured such that a first arriving outbound payload received by the at least one gaming machine is effective to complete the game transaction, irrespective of when and if a second later arriving outbound payload is received by the at least one gaming machine.**

In Mockapetris, there is no "later arriving outbound payload received by the at least one gaming machine", as it <u>only</u> receives a message from the last member of the ring – meaning that there can be no "later arriving outbound payload", as the ring itself is structured so that the original sender only receives a single ACK from the last member of the ring. Therefore, analogizing the claimed embodiment to Mockapetris' "software ring", in effect, breaks the claim and requires ignoring substantial claim limitations for the purpose of making the rejection.

It is respectfully submitted that the Examiner's interpretation of the claimed subject matter relative to Mockapetris' "software ring" cannot reasonably be reconciled with the actual claim language, as forcing such an interpretation requires ignoring positive claim limitations (e.g., "sending a single transaction packet to each" ... "return a single outbound game payload to the gaming machine...", "later arriving outbound payload...", etc.) and the plain meaning of the language of the independent claims.

The **<u>Separate Acknowledgment Algorithm</u>** (number 2 above) relies on one-to-one ACKs, which can result in the cost of acknowledgements being greater than the cost of the message distribution (again, here we are talking about the cost of generating and sending the ACKs, of which the claimed embodiments are agnostic).

The **<u>Saturation Algorithm</u>** (number 3 above), does not even rely on ACKs, but on the transmission of a sufficient number of copies of the message to insure that at least one copy thereof gets through to each destination.

Lastly, the **<u>Negative Acknowledgment Algorithm</u>** (number 4 above) in which members of the multicast that receive the message correctly do not send an ACK and those members of the multicast that would like to be able to copy the message, but cannot, send a NACK.

Each of these algorithms is solely concerned with the manner in which the host generates and sends the ACKs (or not) back to the multicast message sender. The claims, on the other hand, are crafted from the point of view of the receiver of the ACKs (to use Mockapetris' terminology) – that is, from the point of view of the claimed gaming machine, not from the point of view of the central servers that generate and send the outbound payloads (which the Exr. has analogized to Mockapetris' ACKs). None of these algorithms teach or suggest, whether considered alone or in combination with Nguyen (which teaches gaming machines), treating a first arriving outbound payload received by the at least one gaming machine as being effective to complete the game transaction, irrespective of when and if a second later arriving outbound payload is received by the at least one gaming machine, as claimed herein.

It is believed, therefore, that the independent claims define an online gaming systems and methods that find no counterpart in the applied Mockapetris-Nguyen combination. Independent claim 109 shares similar language and is not believed to be obvious over the Mockapetris-Nguyen combination, whether considered alone or in further combination with San Andres. Reconsideration and withdrawal of the outstanding rejections are, therefore, respectfully requested.

Applicants' attorney believes that the present application is now in condition for allowance and passage to issue. If any unresolved issues remain, the Examiner is respectfully invited to contact the undersigned attorney of record at the telephone number indicated below, and whatever is required will be done at once.

Respectfully submitted,

Date: February 3, 2011 By: ______________________

Alan W. Young
Attorney for Applicants
Registration No. 37,970

YOUNG LAW FIRM, P.C.
4370 Alpine Rd., Ste. 106
Portola Valley, CA 94028
Tel.: (650) 851-7210
Fax: (650) 851-7232

C:\YLF\CLIENTS\CYBS\5872 (Univ Gaming Server)\5872 Amendment Responsive to FOA of Oct 13, 2010.doc